# Generalized Neighborhood Attention: Multi-dimensional Sparse Attention at the Speed of Light

Supplementary Material

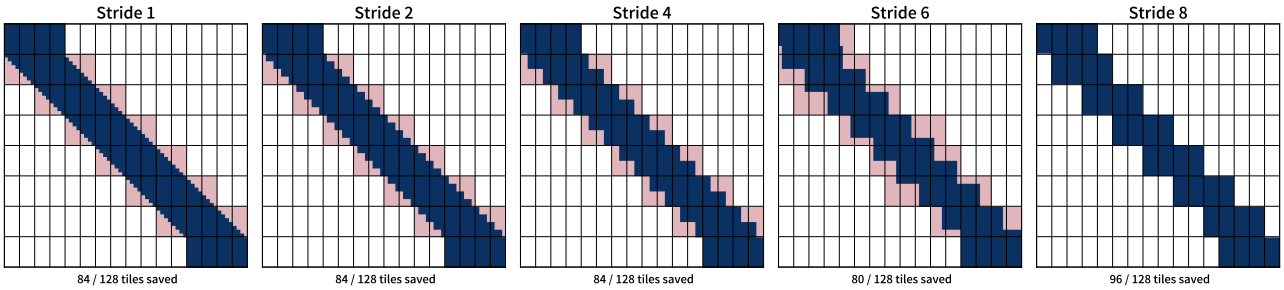



Figure VIII. **Tiles of work saved under different stride values in a 1-D use case.** The use case is a 64-token input, with window size 16, and the kernel is assumed to use a Q tile size of 8, and KV tile size of 4. Stride 8 is the smallest stride that saves more work tiles. It is also the smallest stride that is fully block-sparse.

## A. Effect of stride on quality and efficiency

In this section, we present two visualizations of stride, one demonstrating its effect on block-sparse efficiency (Fig. VIII), and the other its effect on translational equivariance (Fig. I).

Fig. VIII is a visualization of a static block-sparse schedule (block-sparse mask) similar to those in Fig. 3, but only in a single-dimensional use case for simplicity. The use case is a self (neighborhood) attention over 64 tokens, with window size 16, and assumes the query tile size is 8, and KV tile size is 4. As it can be seen, the work tile statistics do not improve under strides 2 through 7, compared to no stride (stride 1), with stride 8 being the first one to result in improved savings. We also note that in this case, stride 8 is also fully block-sparse, meaning none of the computed dot products are masked, and 0 FLOPs are wasted, resulting in the limit of efficiency for this sparsity ratio. However, we also note that the first stride to improve efficiency is not necessarily always fully block-sparse, and both the "helpful strides" and fully block-sparse ones are functions of the exact use case, tile sizes, and other relevant design choices. This is one of the primary motivations behind $\mathcal{N}ATTEN$Sim.

In Fig. I, we visualize 2-D cases, in which we feed a grayscale image through a weightless GNA operation (assuming QKV projections are the identity function), and look at different stride values. Similar to observations made in the original neighborhood attention work [20], we observe the maximum stride, equivalent to blocked / window self attention, introduces a noticeable "patchiness" effect into the result, while the standard neighborhood attention case (stride 1) does not have this problem. What is interesting is that GNA exposes the exact level / size of "patchiness" through the stride parameter. With GNA, end-users designing model architectures will be able to experiment with multiple useful strides and pick the one best trading-off accuracy and efficiency, instead of having to pick either one of the extremes.

## B. Experiment setup

Our experiments were done on H100 and B200 DGX machines, each with 8 GPUs connected via NVLink. However, all of our experiments were limited to only one GPU at a time, and we present no numbers with multi-GPU inference.

Operation-level runtimes were obtained through profiling using the PyTorch profiler, itself using the CUDA Profiling Tools Interface (CUPTI).

Model inference was done using the official codebase from each model, with timers added for measuring runtime. Integration of GNA into each model is further described below.

**Cosmos Predict 1.** We specifically use the 7 billion parameter (7-B) variant, and our chosen workload produces 5 seconds of 720p resolution outputs. This workload involves a token layout shape of $16 \times 44 \times 80$ (56,320 tokens). The default diffusion scheduler uses 35 diffusion steps.

For our Blackwell kernel, we set Q tile shape to $8 \times 4 \times 8$ and KV tile shape to $4 \times 4 \times 8$, which are compatible with the underlying FMHA tile sizes of 256 and 128, and evenly divide the token layout shape. For our Hopper kernel, since the underlying kernel uses tile size 128 for both Q and KV, we simply halve the tile size over the temporal dimension,

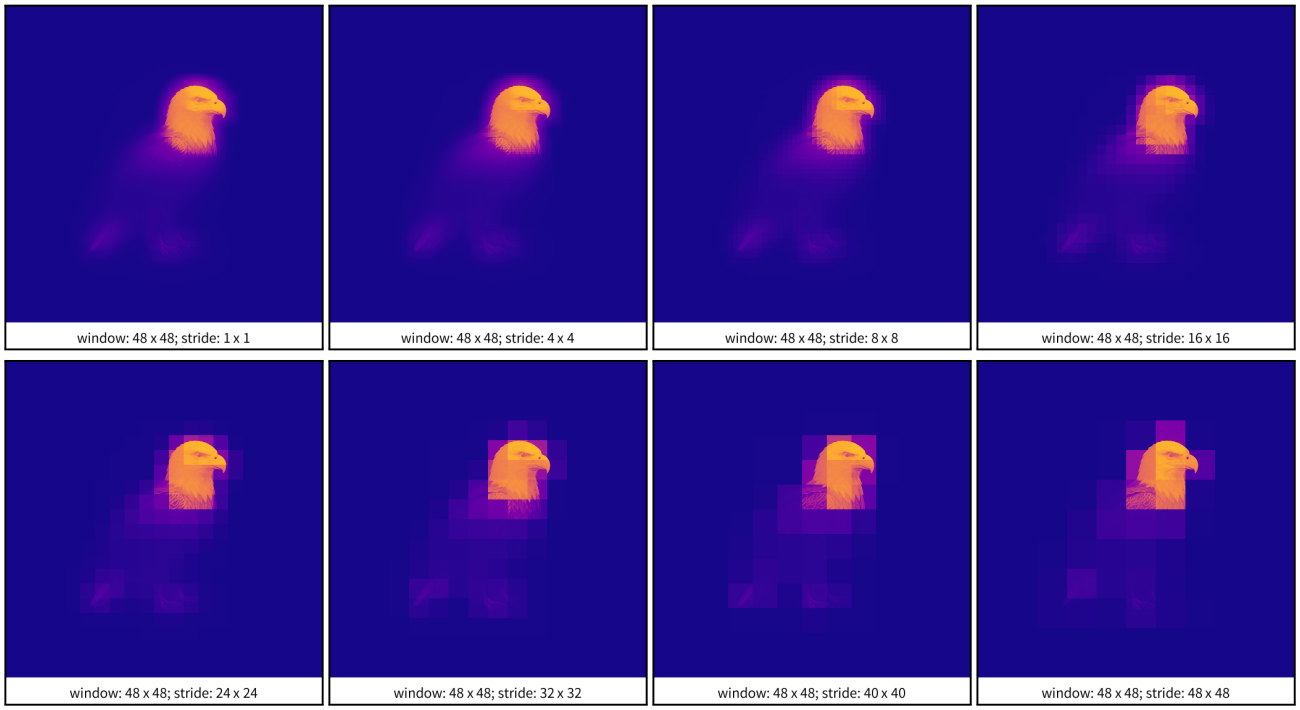



Figure I. **Effect of stride on translational equivariance.** Larger strides introduce a "patch"-like effect, approaching Blocked Attention.

and use 4 × 4 × 8 for both Q and KV, which according to $\mathcal{N}ATTEN$Sim does not affect work tile statistics, and also exhibits similar performance gains as in the Blackwell case. This particular model is very sensitive to sparsity along the temporal dimension, as a result of which we only introduce sparsity along the spatial dimensions. For the medium sparsity setting, we chose window size 16 × 32 × 48, which is approximately 56% sparsity. For non-sparse diffusion steps, we find that retaining the first 12 steps out of 35 provides the best trade-off.

For the high sparsity setting, we chose window size 16 × 24 × 16, which is approximately 89% sparsity. However, as noted earlier, this setting will require further fine-tuning to recover the drop in quality.

Since this model uses a vanilla DiT architecture [34], attention over visual tokens is in a separate layer, making integration only a matter of replacing the self attention operator with GNA with the expected window size and stride.

**Hunyuan Video.** This video generation model uses the MMDiT architecture [17]. The key difference concerning our experiments is that the Self Attention operation over video tokens and Cross Attention between video and text prompt tokens are combined into one Self Attention operator. To apply GNA, we separate the interaction between the different modalities (video and text) into 3 Attention operations. Text tokens cross-attend to both text and video tokens, without any sparsity introduced. Video tokens attend to all text tokens. The Self Attention over video tokens is replaced by GNA. We use attention merging, a feature implemented in $\mathcal{N}ATTEN$, to handle the video attention branch and combine the outputs from GNA and cross attention between video and text tokens. Similar to Cosmos Predict 1, we chose the 5-second 720p video generation use case, which has a token layout shape of 30 × 48 × 80 (115,200 tokens). Unlike Cosmos, this 14-B parameter model runs for 50 diffusion steps.

In the Blackwell FNA kernel, we set Q tile shape to 2 × 16 × 8 and KV tile shape to 2 × 8 × 8, which are compatible with the underlying FMHA tile sizes of 256 and 128, and evenly divide the token layout shape. For the medium sparsity setting, we chose window size 30 × 40 × 40, which is approximately 58% sparsity. For the high sparsity setting, we chose window size 18 × 24 × 24, which is approximately 91% sparsity, but we use Q tile shape 4 × 8 × 8, as $\mathcal{N}ATTEN$Sim finds more balanced stride values for this configuration. For non-sparse diffusion steps, we find that retaining the first 15 steps out of 50 provides the best trade-off. For the Hopper FNA kernel, we simply use tile shape 2 × 8 × 8 for both Q and KV, and for both sparsity settings, even though for the 58% sparsity case, Q tile shape of 1 × 16 × 8 would work slightly better for the fully block-sparse case. This is because the blocked attention and standard NA cases exhibit significantly better performance on Hopper with the

Table I. Cosmos Predict 1 end-to-end speedups from GNA under different sparsity levels and strides. We report both analytical speedups based on $\mathcal{N}ATTEN$Sim, as well as actual speedups on B200. We also report two settings: one with all diffusion steps done with GNA, and the other with the first 12 steps done with self attention. The former can be more representative of cases where the model can be fine-tuned with GNA, while the latter is more applicable for off-the-shelf applications.

| Window Size | Stride | # SA Steps | E2E Speedup ↑ Analytical FLOP-wise | Analytical $\mathcal{N}$ASim | Actual |
|---|---|---|---|---|---|
| **56% sparsity** | | | | | |
| 16 × 32 × 48 | 1 × 1 × 1 | 0 | 1.50× | 1.35× | 1.18× |
| 16 × 32 × 48 | 1 × 8 × 1 | 0 | 1.50× | 1.42× | 1.20× |
| 16 × 32 × 48 | 1 × 1 × 16 | 0 | 1.50× | 1.42× | 1.21× |
| 16 × 32 × 48 | 1 × 8 × 16 | 0 | 1.50× | 1.50× | 1.46× |
| 16 × 32 × 48 | 1 × 1 × 1 | 12 | 1.28× | 1.21× | 1.11× |
| 16 × 32 × 48 | 1 × 8 × 1 | 12 | 1.28× | 1.24× | 1.12× |
| 16 × 32 × 48 | 1 × 1 × 16 | 12 | 1.28× | 1.24× | 1.13× |
| 16 × 32 × 48 | 1 × 8 × 16 | 12 | 1.28× | 1.28× | 1.26× |
| **89% sparsity** | | | | | |
| 16 × 24 × 16 | 1 × 1 × 1 | 0 | 2.10× | 1.90× | 1.76× |
| 16 × 24 × 16 | 1 × 8 × 1 | 0 | 2.10× | 1.96× | 1.79× |
| 16 × 24 × 16 | 1 × 1 × 16 | 0 | 2.10× | 2.05× | 1.88× |
| 16 × 24 × 16 | 1 × 8 × 16 | 0 | 2.10× | 2.10× | 2.05× |
| 16 × 24 × 16 | 1 × 1 × 1 | 12 | 1.52× | 1.45× | 1.40× |
| 16 × 24 × 16 | 1 × 8 × 1 | 12 | 1.52× | 1.48× | 1.40× |
| 16 × 24 × 16 | 1 × 1 × 16 | 12 | 1.52× | 1.51× | 1.44× |
| 16 × 24 × 16 | 1 × 8 × 16 | 12 | 1.52× | 1.52× | 1.50× |

2 × 8 × 8 Q tile shape.

**FLUX.** FLUX [28], like Hunyuan Video [27], also uses the MMDiT architecture [17], for which we use the same approach to introduce GNA into the multi-modal Self Attention operators. FLUX only meets our constraint of being attention-bound when generating 4K images, for which we had to use extra adapters from Yu et al. [46], as FLUX 1.dev does not natively support 4K image generation. In this workload, we end up with token layouts of shape 256 × 256 (65,536 tokens). The default number of diffusion steps is 28.

In our Blackwell FNA kernel, we set Q tile shape to 16 × 16 and KV tile shape to 16 × 8, which are compatible with the underlying FMHA tile sizes of 256 and 128, and evenly divide the token layout shape. For Hopper, we again follow the two prior models and use 16 × 8 as both the Q and KV tile shapes, since the Q and KV tile sizes of the underlying FMHA kernel are both 128. For this model, we only evaluate the high sparsity setting, as we find the quality to be sufficiently similar to the baseline, in addition to the fact that it is the least attention-bound model, shrinking the end-to-end limit for medium level sparsity. For this setting, we chose window size 80 × 80, which is approximately 90% sparsity. For non-sparse diffusion steps, we find that retaining the first 9 steps out of 28 provides the best trade-off.

Table II. Hunyuan Video end-to-end speedups from GNA under different sparsity levels and strides. We report both analytical speedups based on $\mathcal{N}ATTEN$Sim, as well as actual speedups on B200. We also report two settings: one with all diffusion steps done with GNA, and the other with the first 15 steps done with self attention. The former can be more representative of cases where the model can be fine-tuned with GNA, while the latter is more applicable for off-the-shelf applications.

| Window Size | Stride | # SA Steps | E2E Speedup ↑ Analytical FLOP-wise | Analytical $\mathcal{N}$ASim | Actual |
|---|---|---|---|---|---|
| **58% sparsity** | | | | | |
| 30 × 40 × 40 | 1 × 1 × 1 | 0 | 1.55× | 1.21× | 1.15× |
| 30 × 40 × 40 | 1 × 1 × 8 | 0 | 1.55× | 1.44× | 1.26× |
| 30 × 40 × 40 | 1 × 32 × 8 | 0 | 1.55× | 1.55× | 1.53× |
| 30 × 40 × 40 | 1 × 1 × 1 | 15 | 1.33× | 1.14× | 1.08× |
| 30 × 40 × 40 | 1 × 1 × 8 | 15 | 1.33× | 1.27× | 1.15× |
| 30 × 40 × 40 | 1 × 32 × 8 | 15 | 1.33× | 1.33× | 1.30× |
| **91% sparsity** | | | | | |
| 18 × 24 × 24 | 1 × 1 × 1 | 0 | 2.23× | 1.73× | **1.73×** |
| 18 × 24 × 24 | 2 × 1 × 1 | 0 | 2.23× | 1.78× | 1.77× |
| 18 × 24 × 24 | 1 × 1 × 8 | 0 | 2.23× | 1.99× | 1.95× |
| 18 × 24 × 24 | 2 × 1 × 8 | 0 | 2.23× | 2.02× | 1.98× |
| 18 × 24 × 24 | 1 × 8 × 8 | 0 | 2.23× | 2.17× | 2.09× |
| 18 × 24 × 24 | 2 × 8 × 8 | 0 | 2.23× | 2.20× | 2.11× |
| 18 × 24 × 24 | 16 × 8 × 8 | 0 | 2.23× | 2.23× | **2.23×** |
| 18 × 24 × 24 | 1 × 1 × 1 | 15 | 1.63× | 1.42× | **1.42×** |
| 18 × 24 × 24 | 2 × 1 × 1 | 15 | 1.63× | 1.44× | **1.44×** |
| 18 × 24 × 24 | 1 × 1 × 8 | 15 | 1.63× | 1.53× | 1.51× |
| 18 × 24 × 24 | 2 × 1 × 8 | 15 | 1.63× | 1.55× | 1.54× |
| 18 × 24 × 24 | 1 × 8 × 8 | 15 | 1.63× | 1.61× | 1.58× |
| 18 × 24 × 24 | 2 × 8 × 8 | 15 | 1.63× | 1.62× | 1.59× |
| 18 × 24 × 24 | 16 × 8 × 8 | 15 | 1.63× | 1.63× | **1.63×** |

## C. Additional experiment results

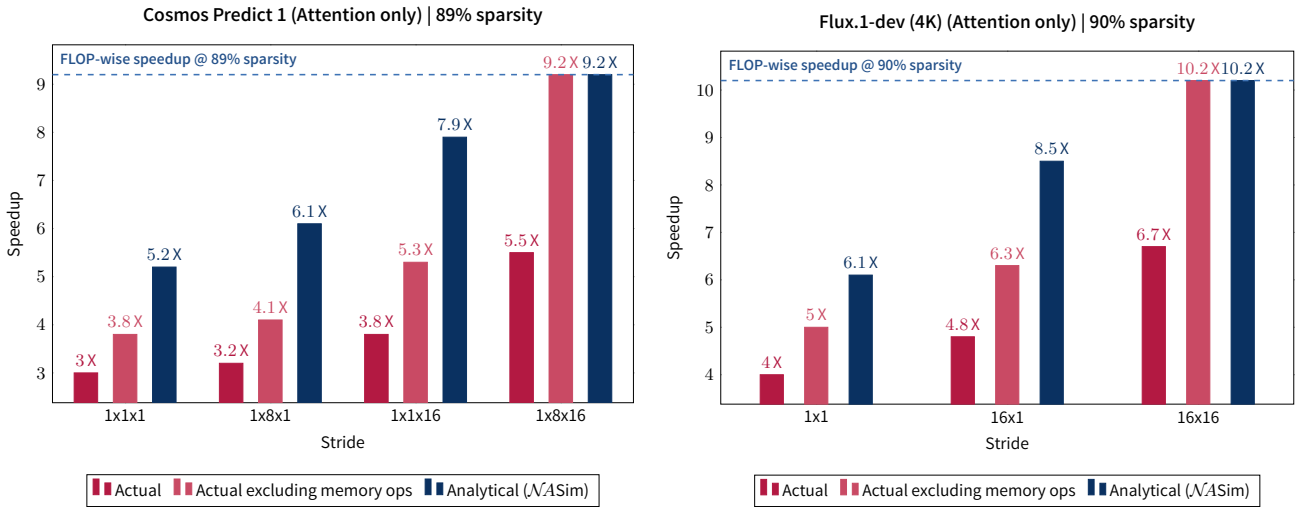
In this section, we present additional experiment results excluded from the main text for brevity and due to the page limit.

**Operation-level speedups.** We present speedups from the attention operator over the Cosmos Predict 1 [1] and FLUX [28] on the B200 in Fig. II. This is a similar setting to Fig. 6, which presented the Hunyuan Video [27] case.

**End-to-end speedups.** We present extended versions of Tabs. 2 to 4 in this section, showing the measurements for more sparsity configurations.

In Tab. I, we present both the medium and high sparsity settings for Cosmos Predict 1 [1], and both with and without retention of self attention steps. Even though we mainly used the medium sparsity setting with retention, this was only done to preserve quality without fine-tuning. Further fine-tuning will enable further sparsity increases, which means if fine-tuned, returns for Cosmos Predict 1 could be as great as 2.05× (high sparsity, no SA retention).

In Tab. II, we present both sparsity settings with and without SA retention for Hunyuan Video [27]. Unlike Cosmos,




(a) Cosmos Predict 1 use case with 89% sparsity (window size $16 \times 24 \times 16$). (b) FLUX.1-dev (4K) use case with 90% sparsity (window size $80 \times 80$).

Figure II. Operation-level (attention only) speedups on Cosmos and FLUX with approximately 90% sparsity through GNA. Analytical speedup is according to $\mathcal{NATTEN}$Sim, and actual speedups are measured by running on B200. Note that with the perfectly block-sparse strides, our decomposed FNA implementation can match the full analytical speedup, but can become limited by the naive implementation of the memory operation (token permute).

Hunyuan can tolerate high sparsity, but again mainly only with retention, with a speedup of 1.63× on the B200. However, if fine-tuned with this setting, Hunyuan Video could see a larger 2.23× speedup. We further note that this setting, while not our first choice for preserving quality, is the same used by STA [49] in their VBench [23] evaluation, to which we compared our work in Tab. 5. The speedups reported in that table were for the Hopper architecture, since that is the only architecture STA supports. We present the evaluation but with our Blackwell speedups in Tab. IV.

Finally, in Tab. III, we present FLUX [28] numbers with and without retention, since we only evaluated FLUX with high sparsity. On FLUX we achieved a 1.45× with SA retention, but further fine-tuning could boost that up to 1.82×, which is the speedup without retention.

## D. Additional qualitative results

We finally present additional samples from each model in Figs. III to V.

Table III. FLUX-1.dev with URAE [46] end-to-end speedups from GNA under different strides. We report both analytical speedups based on $\mathcal{NATTEN}$Sim, as well as actual speedups on B200. We also report two settings: one with all diffusion steps done with GNA, and the other with the first 9 steps done with self attention. The former can be more representative of cases where the model can be fine-tuned with GNA, while the latter is more applicable for off-the-shelf applications.

| Window Size | Stride | # SA Steps | E2E Speedup ↑ Analytical FLOP-wise | Analytical $\mathcal{NA}$Sim | Actual |
|---|---|---|---|---|---|
| $80 \times 80$ | $1 \times 1$ | 0 | 1.88× | 1.76× | 1.65× |
| $80 \times 80$ | $16 \times 1$ | 0 | 1.88× | 1.84× | 1.72× |
| $80 \times 80$ | $16 \times 16$ | 0 | 1.88× | 1.88× | 1.82× |
| $80 \times 80$ | $1 \times 1$ | 9 | 1.46× | 1.42× | 1.37× |
| $80 \times 80$ | $16 \times 1$ | 9 | 1.46× | 1.45× | 1.40× |
| $80 \times 80$ | $16 \times 16$ | 9 | 1.46× | 1.46× | 1.45× |

Table IV. Hunyuan Video VBench configuration performance on Blackwell across different GNA configurations. Following STA [49], these configurations do not have any self attention steps. In this setting, we followed STA [49] and applied sparsity to all diffusion 50 steps, as opposed to performing self attention for the first few steps, which increases the percentage of the workload we accelerate and by extension end-to-end speedups. We do not report model FLOPs, as it is a poor measure of efficiency, and instead report two analytical speedups: one based on FLOPs, which would be the same across methods for any given attention sparsity, and another based on $\mathcal{N}ATTEN$Sim. We also report actual speedup based on GNA runtimes, which use our Blackwell FNA kernel. However, note that the above runtimes can vary $\pm$ 3.5%. We do not report end-to-end speedups from STA, as our study was done on the Blackwell architecture, and STA's implementation was specific to the Hopper architecture, and the difference in workload distribution would skew the comparison. We further note it is not possible to run kernels that target Hopper Tensor Cores on any architecture other than Hopper. In addition, STA's step size of $6 \times 8 \times 8$ by definition requires tile size 384 for both Q and KV, which is not implementable with the current Blackwell FMHA, and therefore our implementation. We finally point out that with a speedup of approximately 2.23× in the 91% sparsity case, **GNA achieves the maximum speedup theoretically possible** for this exact level of sparsity (with respect to reduction in FLOPs).

| **Method** | **Window Size** | **Stride** | **Attention Sparsity** | **VBench** | | | **Runtime ↓ on B200 (seconds)** | **End-to-End Speedup ↑** | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | | | **Total ↑** | **Quality ↑** | **Semantic ↑** | | **Analytical** | | **Actual** |
| | | | | | | | | **FLOP-wise** | **$\mathcal{N}$ASim** | |
| **SA** (baseline) | - | - | 0.0% | 83.08% | 85.01% | 75.35% | 628 | | | |
| ***As reported in STA [49]*** | | | | | | | | | | |
| **Tiled NATTEN** (= NA) | 30 × 41 × 41 | - | 58.3% | 82.69% | 84.61% | 75.00% | - | 1.55× | - | - |
| **Swin** (= WSA) | 30 × 40 × 40 | - | 58.3% | 77.53% | 78.84% | 72.28% | - | 1.55× | - | - |
| **STA** | 30 × 40 × 40 | - | 58.3% | 82.46% | 84.63% | 73.83% | - | 1.55× | - | - |
| **STA** | 18 × 24 × 24 | - | 91.0% | 80.58% | 81.47% | 77.03% | - | 2.23× | - | - |
| **STA w/ training** | 18 × 24 × 24 | - | 91.0% | 82.62% | 84.76% | 74.05% | - | 2.23× | - | - |
| ***Ours*** | | | | | | | | | | |
| **GNA** (= NA) | 30 × 40 × 40 | 1 × 1 × 1 | 58.3% | 83.24% | 84.70% | 77.42% | 546 | 1.55× | 1.21× | 1.15× |
| **GNA** (= WSA) | 30 × 40 × 40 | 30 × 40 × 40 | 58.3% | 82.25% | 83.11% | 78.83% | 491 | 1.55× | 1.44× | 1.28× |
| **GNA** | 30 × 40 × 40 | 1 × 32 × 8 | 58.3% | 83.40% | 84.77% | 77.92% | 411 | 1.55× | 1.55× | 1.53× |
| **GNA** (= NA) | 18 × 24 × 24 | 1 × 1 × 1 | 91.0% | 82.36% | 83.03% | 79.68% | 359 | 2.23× | 1.73× | **1.73×** |
| **GNA** (= WSA) | 18 × 24 × 24 | 18 × 24 × 24 | 91.0% | 80.25% | 80.97% | 77.38% | 314 | 2.23× | 2.06× | 2.00× |
| **GNA** | 18 × 24 × 24 | 16 × 8 × 8 | 91.0% | 82.04% | 82.62% | 79.69% | 277 | 2.23× | 2.23× | **2.23×** |

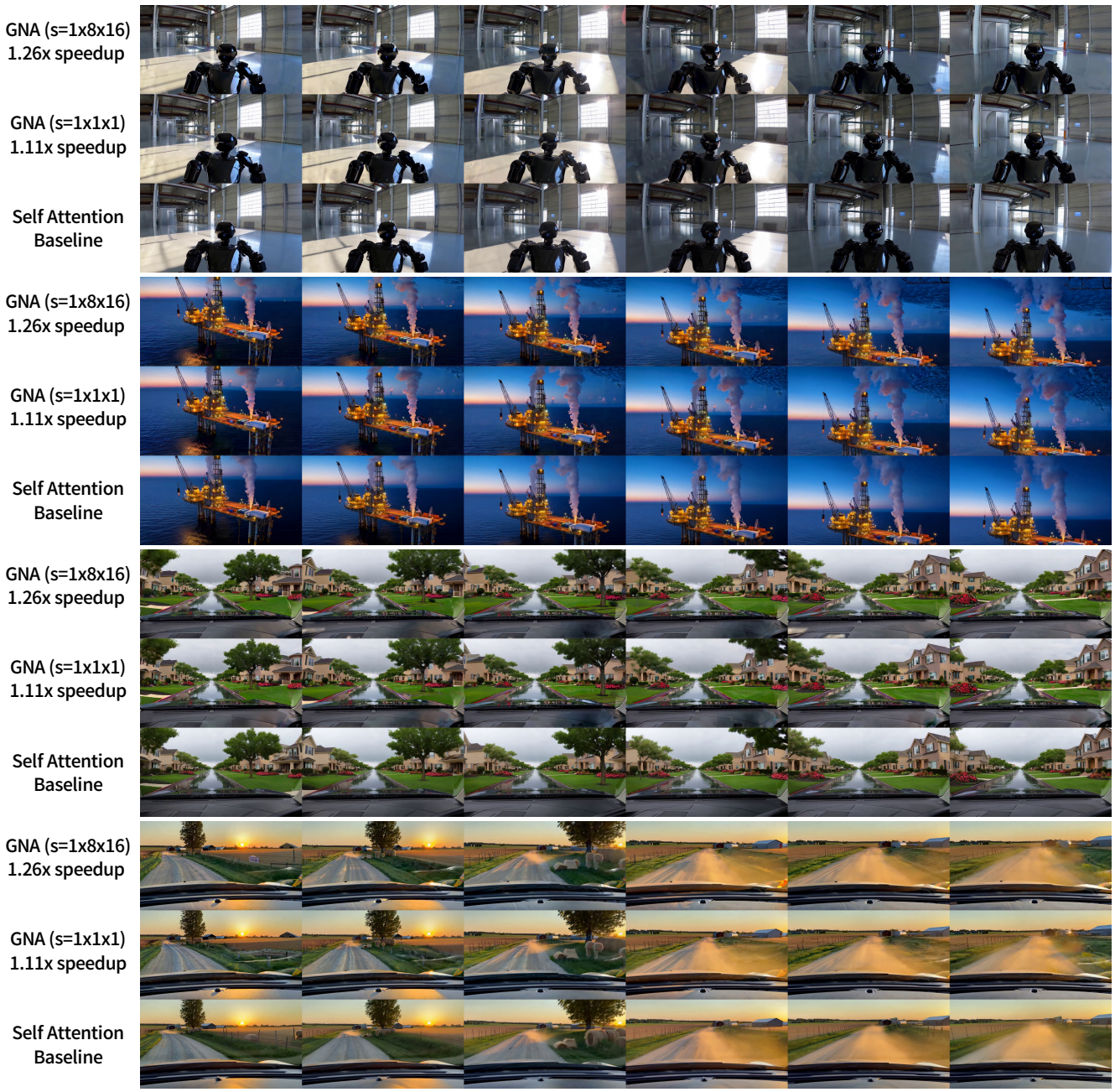



Figure III. Sample frames from videos generated by Cosmos Predict 1, with GNA introduced into the last 23 of the 35 diffusion steps. Window size is $16 \times 32 \times 48$ ($\approx 56\%$ sparsity). Speedup limit under this setting, with the same level of sparsity, is 1.28×.

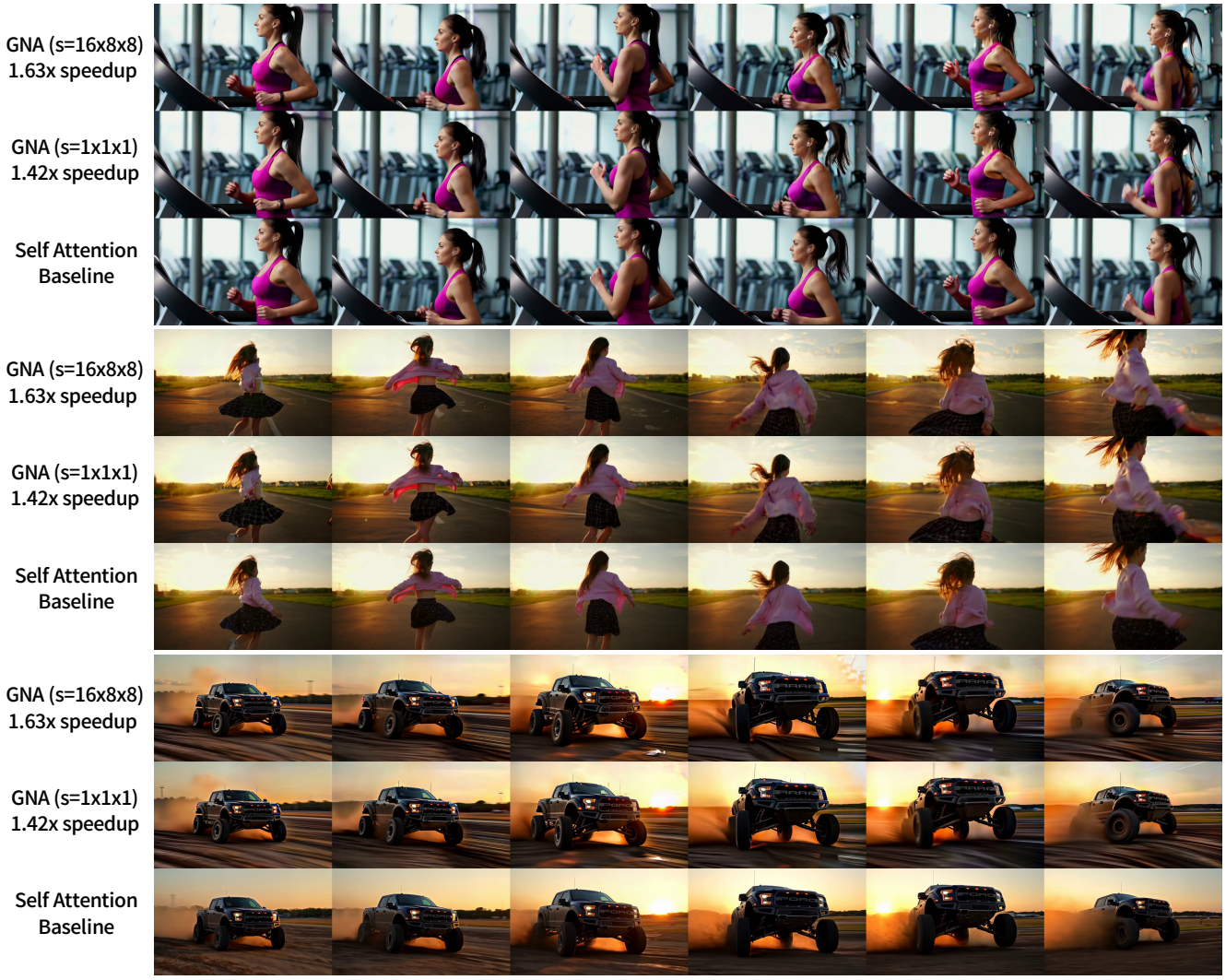



Figure IV. Sample frames from videos generated by Hunyuan Video, with GNA introduced into the last 35 of the 50 diffusion steps. Window size is $18 \times 24 \times 24$ ($\approx 91\%$ sparsity). Speedup limit under this setting, with the same level of sparsity, is 1.63×.

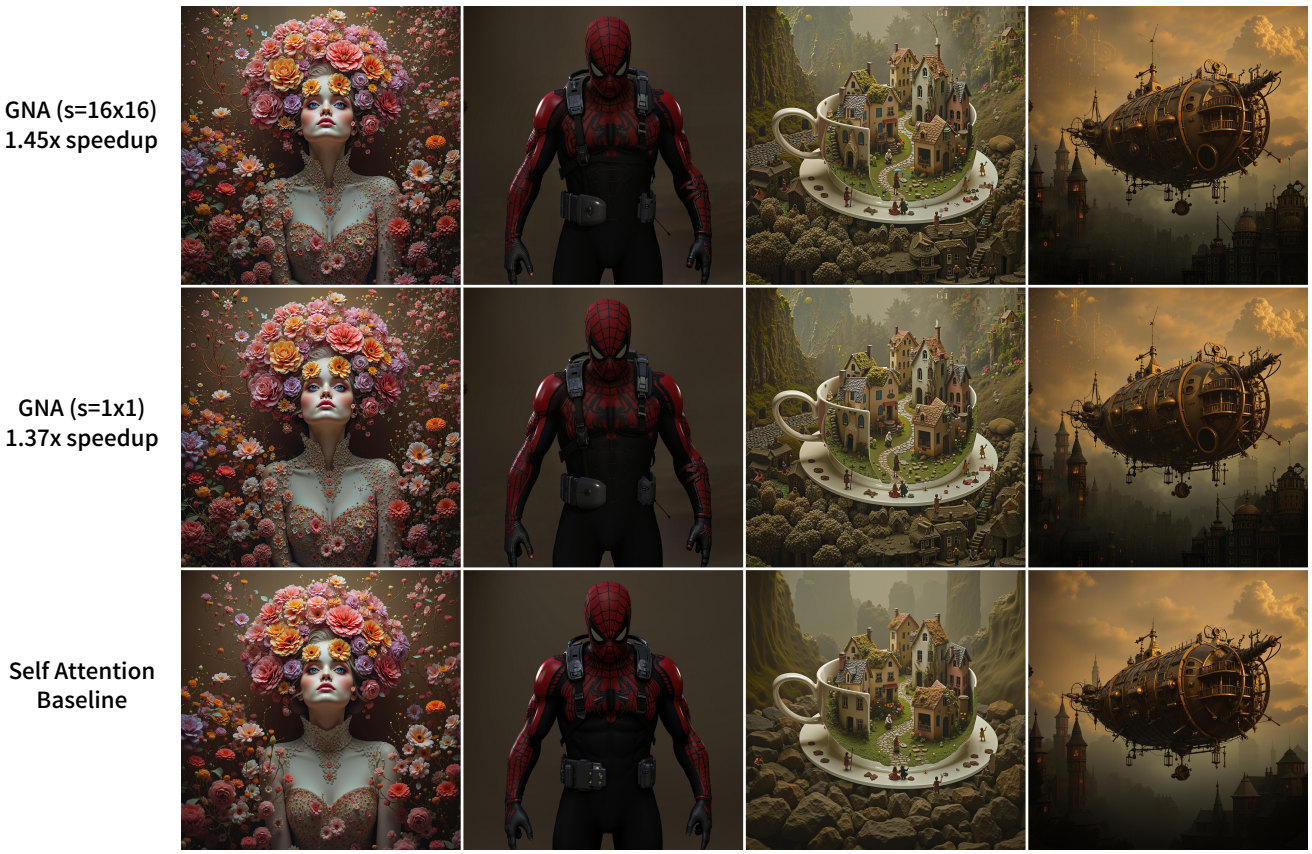



Figure V. Images generated by ultra-resolution FLUX [28, 46] with GNA introduced into the last 19 of the 28 diffusion steps. Window size is $80 \times 80$ ($\approx 90\%$ sparsity). Speedup limit under this setting, with the same level of sparsity, is 1.46×.